Project Requirements

Every project should meet the following requirements. We will discuss a few aspects of these requirements in more detail during the early weeks of working on the project (these are marked with * stars). If you're ever unsure of what something here specifically requires, just ask!

Functionality

The application should be functional, with a full set of completed features as laid out in your user stories.

Teamwork

All teams will:

- Have a leader position that rotates through the members of the team, changing each week. The leader is responsible for tracking and coordinating work during their tenure as leader, and they will submit the iteration report at the end of each iteration. (If a team decides together to assign a leader in a different fashion, they need to check with me first.)
- Hold at least one meeting per week outside of class to discuss progress, make plans, share and explain code, and do whatever else helps make progress as a team.
- Select and use some form of group messaging (slack, groupme, texts, or otherwise).

Github

Code must be hosted on Github. Commits should be logical and well-structured, tracking changes to the code over time, explained with meaningful commit messages, showing which team member developed what, etc.

Teams should track user stories to be completed, bugs to be fixed, and other tasks using Github Issues. When a task is assigned to a team member, the corresponding issue should be assigned to that person. Use comments within each issue to ask questions, discuss details, and keep track of any decisions made about it over time.

Your team should use the "Github flow" for using Git as a team. This means using separate branches (separate from main) for each new feature/bugfix/enhancement, and uses pull requests to merge them back into main. Every pull request must be reviewed by at least one other team member (other than the one who authored it). See Using Git in a Team.

Testing

All Python code should be tested by a suite of unit tests.

Every person who develops a new feature should also write and commit unit tests for that feature at the same time. Every pull request adding a feature should contain unit tests for it (if applicable. If you have questions about this, ask Daniel or Anna).

Accessibility

The application's page(s) should be accessible, following the accessibility guidelines we studied early in the semester.

User Interface *

The UI should be well-designed, following the guidelines for good, usable user interface design. It should be attractive, clean, consistent, clear, understandable, and overall *usable*.

Security *

The application should be free of any security issues we discuss in class. This includes being free of SQL injection vulnerabilities and storing passwords securely.

** means we will be covering this in more detail as the project progresses.*